MIDI Capability Inquiry (MIDI-CI)

Document version 1.0
April 26, 2018

Published by:
The MIDI Manufacturers Association
Los Angeles CA

PREFACE

MIDI has been a successful tool for more than 3 decades. The features of MIDI 1.0 continue to work well. The basic semantic language of music does not change and as a result the existing definitions of MIDI as musical control messages continue to work remarkably well.

However, MIDI has not changed to fully take advantage of the new technical environment around it. We want to expand the feature set of MIDI capabilities.

At the same time, we recognize there are several key hurdles and requirements to consider as we make any additions to MIDI:

- Backwards compatibility is a key requirement. Our users expect new MIDI devices to work seamlessly with MIDI devices sold over the past 33 years.
- All MIDI Status Bytes are defined. The opcodes and data payloads are defined. It is difficult to define any new message types or change the format of the existing MIDI messages.

Expanding MIDI with new features requires a new protocol with extended MIDI messages. To protect backwards compatibility in an environment with expanded features, devices need to confirm the capabilities of other connected devices. When 2 devices are connected to each other, they use MIDI 1.0 and confirm each other's capabilities before using expanded features. If both devices share support for the same expanded MIDI features they can agree to use those expanded MIDI features. MIDI-CI provides this mechanism.

MIDI-CI: Solution for Expanding MIDI while Protecting Backwards Compatibility:

MIDI Capability Inquiry (MIDI-CI) is a mechanism to allow us to expand MIDI with new features while protecting backward compatibility with MIDI devices that do not understand these newly defined features.

MIDI-CI separates older MIDI products from newer products with new capabilities and provides a mechanism for two MIDI devices to understand what new capabilities are supported.

MIDI-CI assumes and requires bidirectional communication. Once a MIDI-CI connection is established between devices, query and response messages define what capabilities each device has. MIDI-CI then negotiates or auto-configures to use those features that are common between the devices.

MIDI-CI provides test mechanisms when enabling new features. If a test fails, then devices fall back to using MIDI 1.0 for that feature.

MIDI-CI improves MIDI capabilities in several key areas. MIDI-CI allows devices to use an expanded MIDI protocol with high resolution and multiple per note controllers. It allows for incremental adoption of new MIDI features by providing a fallback to MIDI 1.0 devices in all cases.

MIDI-CI Includes Queries for 3 major areas of expanded MIDI functionality:

1. Protocol Negotiation
2. Profile Configuration
3. Property Exchange

CA-035 MIDI Capability Inquiry Specification

MMA
PO BOX 3173
La Habra CA 90632-3173
USA

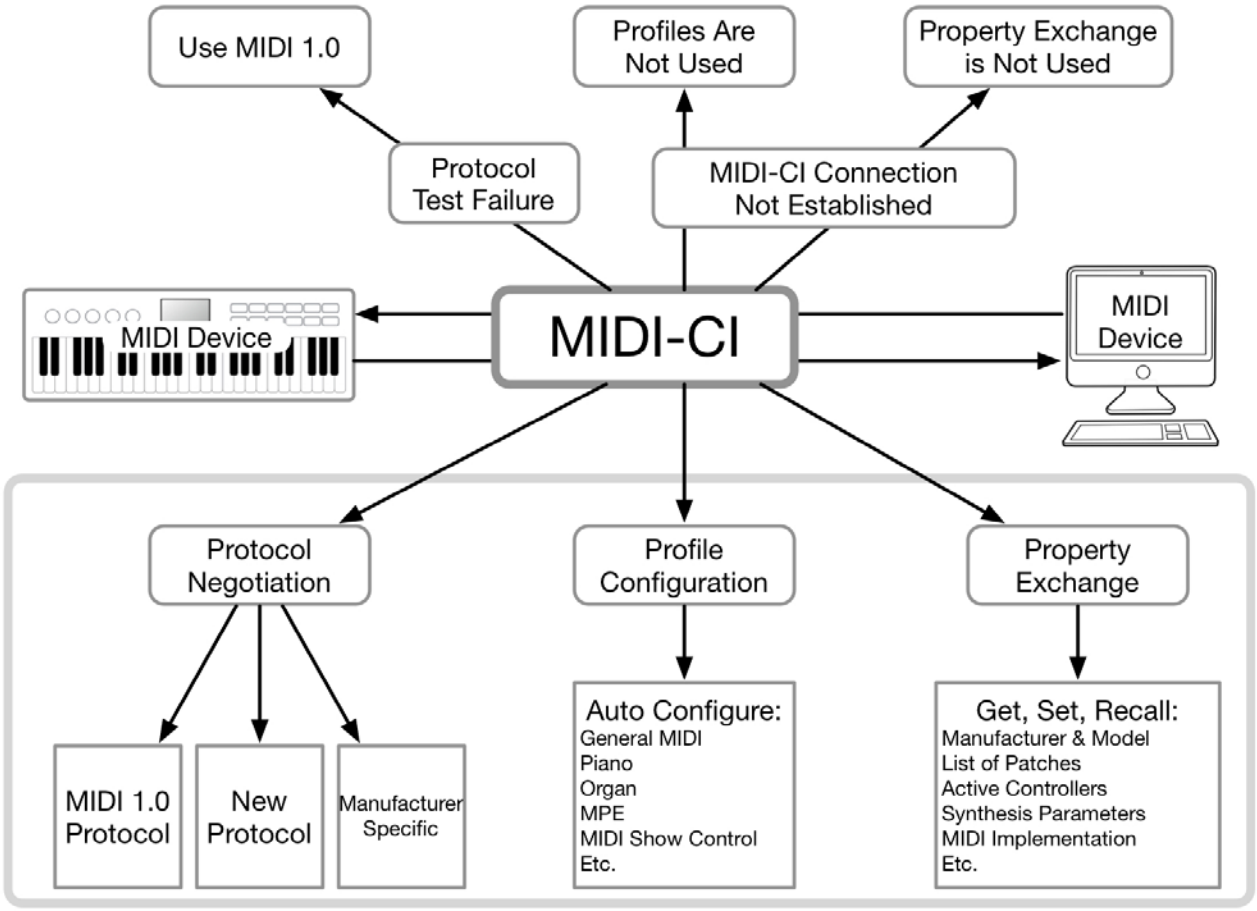
Table of Contents

1. INTRODUCTION

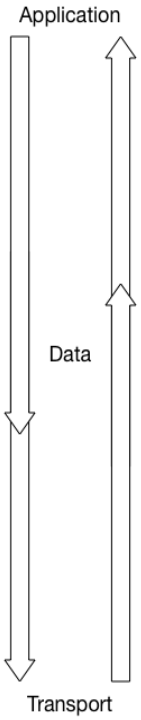
1.1 Background

MIDI-CI defines an architecture that allows devices with bidirectional communication to agree to use extended MIDI capabilities beyond those defined in MIDI 1.0, while carefully protecting backward compatibility. MIDI-CI features “fall back” mechanisms so that if a device does not support new features MIDI continues to work as defined by MIDI 1.0. Goals of MIDI-CI design:

1. Fully backward compatible: supports continued MIDI 1.0 functionality for any devices that do not recognize extended MIDI features enabled by MIDI-CI.
2. Allow easy configuration between MIDI-CI Devices.
3. Sender can know the capabilities of a Receiver.
4. Sender and Receiver can negotiate auto-configuration details.
5. Define method for negotiating choice of Protocol between devices.
6. Define method for using Profiles.
7. Define method for Discovering, Getting, and Setting a wide range of device Properties.



Layer Model for MIDI-CI



MIDI-CI INQUIRY & NEGOTIATION	MIDI LAYER	DEFINITION	EXAMPLES	MIDI-CI FUNCTION
PROPERTY EXCHANGE	**DEVICE**	Manufacturer and Model Specific Details	Products sold to the users of MIDI	MIDI-CI allows get and set for a wide range of properties or state of a device.
	MIDI IMPLEMENTATION	Channels and Messages Supported	Replaces original "MIDI Implementation Chart"	MIDI-CI allows discovery of MIDI implementation details of a device.
PROFILE CONFIG-URATION	**PROFILES**	A collection of defined Device Parameters and MIDI Messages common across manufacturers	General MIDI, MPE, Hi-Res Piano Profile (to come soon)	MIDI-CI allows general auto-configuration of MIDI Implementation between MIDI devices that share common Profiles.
PROTOCOL NEGOTIATION	**PROTOCOL**	Data Language (MIDI Messages with 1 Status Bit and 7 Data Bits)	Note-On, Control Change, Program Change	MIDI-CI allows selection of MIDI 1.0 messages or a "New AMEI/MMA Protocol" with higher resolution or with timestamps, or added functionality, or other protocols.
none	**BANDWIDTH**	Data Flow Rate, Throughput, Speed	31.25Kbs on 5pinDIN, 31.25Kbs on USB, 1Mbs on USB	None (handled by Transport)
none	**CONNECTION**	Hardware or Software Connection	5PinDIN cable, USB, New 2 Way UART, Ethernet, OS API, VST, CoreMIDI	None (handled by Transport)

Related Documents:

1. MIDI 1.0 Detailed Specification, Document Version 4.2 September 1995
2. USB Device Class Definition for MIDI Devices, Version 1.0
3. Specification for MIDI over Bluetooth Low Energy, Version 1.0
4. RTP Payload Format for MIDI (IETF RFC 6295)

Future Documents:

MIDI-CI defines a foundational structure for expanding MIDI. It does not define the expansions themselves. The expansions of MIDI that MIDI-CI enables will be defined in future documents of the MMA and AMEI. These documents may include the following and more:

1. New AMEI/MMA Protocol Specification - Adds new features to existing MIDI messages and defines new MIDI messages.
2. Common Rules for MIDI Profiles - Defines rules for all Profile Specifications
3. Individual Profile Specifications - Define implementation requirements for compliant devices
4. Property Exchange Data Specification - Defines Property Data semantics for Property Exchange
5. Approved Properties - Define Properties with range of values used by Property Exchange

1.2 Terminology

Initiator - 1 of 2 MIDI devices with a bidirectional communication between them. The Initiator is the device that sends an Inquiry message to a Responder to start a MIDI-CI Session with expectation of reply from a Responder. Initiator has the management role of setting and negotiation parameters for interoperability between the 2 devices. The primary goal of Initiator is usually (but not strictly required to be) configuring 2 devices for subsequent communication from Initiator as MIDI transmitter to Responder as MIDI receiver.

MIDI Cable - a physical transport cable capable of carrying 1 MIDI Stream.

MIDI-CI Device - A device that has the ability to act as a Responder that replies to inquiries received from an Initiator. The ability to act as an Initiator is recommended but optional.

MIDI Endpoint - Original Source of MIDI Messages or Final Consumer of MIDI Messages**.** Supports only MIDI 1.0 or Switchable between MIDI 1.0 and New AMEI/MMA Protocol messages.

MIDI Event Processor (Sequencer, Arpeggiator) - Records, Edits, and Plays messages or Transforms messages in real time**.** Supports only MIDI 1.0 or Switchable between MIDI 1.0 and New AMEI/MMA Protocol messages on a port by port basis.

MIDI Gateway – A special purpose embedded OS device (e.g., Workstation, Router) that manages a Node of Connected Devices/Applications/Plugins. Provides connection & routing between all Endpoints on the Node. Acts as MIDI-CI Proxy for Endpoints whenever necessary. Supports only MIDI 1.0 or Switchable between MIDI 1.0 and New AMEI/MMA Protocol messages.

MIDI Node Server (PC) - General purpose OS, with wide range of MIDI service and MIDI API e.g., Mac or Windows PC. Manages a Node of Connected Devices/Applications/Plugins. Provides connection & routing between all Endpoints on the Node. Acts as MIDI-CI Proxy for Endpoints whenever necessary. May act as MIDI-CI Proxy for Single Direction Endpoints (i.e. API as Proxy for Plugin) Supports both MIDI 1.0 and New AMEI/MMA Protocol messages. It is strongly recommended that a Central MIDI Node PC have protocol translation capability on every Input/Output MIDI-CI connection.

MIDI Port - a physical connector associated with a MIDI Endpoint. Some people may consider a MIDI Port to be synonymous with a MIDI Endpoint. A MIDI Port always has a MIDI Endpoint. But a MIDI Endpoint may not always have a (physical) MIDI Port; It may have a Virtual MIDI Port instead. Inside software a MIDI Port is a virtual representation of a MIDI Port. In this case it probably should be called a Virtual MIDI Port.

MIDI-CI Proxy - a MIDI-CI device, such as a MIDI Node Server, MIDI Gateway, or MIDI Translator, that represents another MIDI device in a MIDI-CI session. While acting as a MIDI-CI Proxy, a device reports the manufacturer and other fields with values from another device that it is representing and not its own.

MIDI-CI Session - A set of MIDI-CI messages exchanged between Initiator and Responder devices to perform a set of Inquiries, Replies and subsequent Negotiation. The MIDI-CI Session starts when the first MIDI-CI message is sent from an Initiator to a Responder to establish a bidirectional connection. The same MNID is used throughout a MIDI-CI Session. The MIDI-CI Session ends when the devices are disconnected.

MIDI Translator - Located between Bidirectional Endpoints**.** Performs Translation between MIDI 1.0 and New AMEI/MMA Protocol whenever necessary. Acts as MIDI-CI Proxy for Endpoints whenever necessary. Passes both MIDI 1.0 and New AMEI/MMA Protocol messages.

MIDI Transport – Carries MIDI data between Bidirectional Endpoints**.** Acts as MIDI-CI Proxy for Endpoints whenever necessary. Passes both MIDI 1.0 and New AMEI/MMA Protocol messages. Does NOT do any protocol translation.

MNID MIDI Negotiation Identifier - A 28bit random number generated by an Initiator to create a clear connection between 2 devices. Initiator and Responder use the same MNID in all MIDI-CI messages throughout all stages of a MIDI-CI Session.

Negotiation - An exchange of MIDI-CI messages managed by an Initiator device to configure Initiator and Responder devices for improved interoperability.

Profile - a set of MIDI messages and defined responses to those messages. General MIDI is one example of a profile (although in original form it does not quite meets all requirements of a MIDI-CI Profile). A Profile may have a defined minimum set of messages that must be supported, along with some optional or recommended messages. General MIDI defines a requirement for supporting GM messages on all 16 channels of 1 virtual cable (or stream). Future Profiles may define support on only 1 channel within a virtual cable, or support for more than 16 channels using multiple virtual cables or Groups.

Protocol - A defined data message structure that defines the semantics for MIDI control messages. In MIDI 1.0 the Protocol includes addressing in the form of 16 MIDI Channels, the Opcode of the control message being sent, and for some message types a value associated with that Opcode.

Responder - 1 of 2 MIDI devices with a bidirectional communication between them. The Responder is the device that receives an Inquiry message from an Initiator device as part of a MIDI-CI Session and acts based on negotiation messages managed by an Initiator device.

Single Direction Endpoint – A MIDI device that only sends or receives and is, therefore, NOT capable of MIDI-CI. In some cases, a MIDI Node Server, MIDI Gateway, MIDI Translator, or MIDI Transport may "know" details about a device from a non-MIDI source (USB Device Descriptors, PlugIn API properties, etc.). The MIDI Node Server, MIDI Gateway, MIDI Translator, or MIDI Transport may act as a MIDI-CI Proxy: it engages in MIDI-CI negotiation in place of the Single Direction Endpoint device.

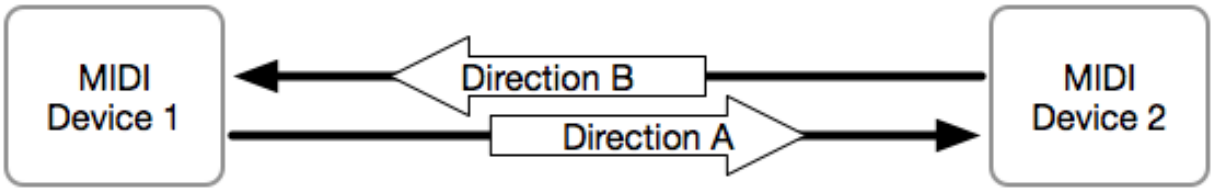
USB Endpoint - the source or sink of data sent over USB. This is a physical device with a buffer. A typical USB-MIDI Interface has 3 USB Endpoints: 1 bidirectional Control Endpoint, and 2 USB-MIDI Endpoints (one in each direction) that each can support up to 16 virtual cables with 16 channels each (256 channels).

USB-MIDI Endpoint - a USB Endpoint used to transfer MIDI Data. In version 1.0 of the USB-MIDI specification, a USB Endpoint supports 16 virtual cables worth of MIDI data, by using a 32bit MIDI Event packet.

2. TOPOLOGY

2.1 Bidirectional

MIDI-CI requires bidirectional MIDI communications.



Every MIDI-CI capable Input Port MUST be paired with a matching Output Port. Devices may have multiple pairs of MIDI Ports for Input and Output.

Each pair of Ports (1 Input + 1 Output) negotiates an independent MIDI-CI session. MIDI-CI uses a MIDI Negotiation Identifier "MNID", a 28 bit random number to track the pairing on matching Input and Output Ports.

See Appendix A for topology limitations related to MIDI Thru.

2.2 Initiator and Responder Relationship

In a MIDI-CI bidirectional connection, both devices are sender and both devices are receiver for various MIDI-CI messages. Therefore, Initiator and a Responder are terms used to clarify the relationship between the 2 MIDI devices.

An Initiator and a Responder are 2 MIDI-CI capable devices with a bidirectional connection. Either of the 2 devices can choose to function as the Initiator. The Initiator takes on the management role of setting and negotiation of parameters for interoperability between the 2 devices. The Initiator starts a MIDI-CI Session by sending an Inquiry message.

The Responder is the device that receives an Inquiry message as part of a MIDI-CI Session and acts based on negotiations managed by an Initiator device.

Some MIDI-CI negotiations make settings that are common to both directions. In these cases, when the Initiator determines and controls negotiation of particular settings, it does so for equal and identical subsequent interoperability in both directions.

Some MIDI-CI negotiations make settings for chosen interoperability in just one direction. In these cases, when the Initiator determines and controls negotiation of particular settings, it does so for subsequent interoperability from Initiator to Responder.

2.3 Bidirectional Negotiation for Bidirectional Settings

MIDI-CI negotiations for Protocol cause simultaneous changes to the protocol used in both directions. In the diagram above, regardless of whether MIDI Device 1 or MIDI Device 2 is the initiator, MIDI-CI negotiation establishes a chosen protocol in both Direction A and Direction B with just one negotiation session.

2.4 Bidirectional Negotiation for Single Direction Settings

MIDI-CI negotiations for Profile Configuration and Property Exchange are intended for discovering and making settings for interoperability in just one direction, from Initiator to Responder.

When a device chooses to be Initiator, it takes on the management role of setting parameters for MIDI interoperability from Initiator to Responder. If the device that is Responder also wants to set parameters in the opposite direction, then the Responder switches roles and becomes Initiator to manage settings for communication in the opposite direction.

> Example: If MIDI Device 1 starts as Initiator Device by sending a Property Exchange Inquiry Has Property and the MIDI Device 2 acts as Responder and answers with a Property Exchange Has Property = True, then MIDI Device 1 may Set that Property on MIDI Device 2. If then the MIDI Device 2 wishes to Set a Property on MIDI Device 1 (that is the opposite direction), then it must then take on the role of Initiator to start another MIDI-CI Property Exchange inquiry.

3. MIDI-CI COMMON RULES AND GUIDELINES

This section outlines concepts and rules common to all Inquiry and Negotiations.

MIDI-CI defines messages and mechanisms for Inquiry and Negotiation in the following categories of MIDI-CI Inquiry and Negotiation of device capabilities:

1. Protocol Negotiation
2. Profile Configuration (includes Manufacturer and Model Information)
3. Property Exchange

3.1 Initiator and Responder Relationship

An Initiator and a Responder are 2 MIDI devices with a bidirectional connection.

MIDI-CI assumes that communication should tend to be receiver centric; it assumes a system where a MIDI transmitter “learns” something about the receiver and adapts its output as much as possible to support the capabilities of the receiver. Through MIDI-CI negotiation mechanisms, a transmitter can also ask a receiving device to enable features reported as supported capabilities to adapt the Receiver to the capabilities of the Sender.

The Initiator has the management role of setting and negotiation parameters for interoperability between the 2 devices. The primary goal of Initiator is usually (but not strictly required to be) configuring 2 devices for subsequent communication from Initiator as MIDI transmitter to Responder as MIDI receiver.

The Responder is the device that receives an Inquiry message as part of a MIDI-CI Session and acts based on negotiations managed by an Initiator device.

The Initiator starts a MIDI-CI Session by sending an Inquiry message.

Initiating MIDI-CI Session with Inquiry and Negotiation

A MIDI-CI Session can be started by several different events on the Initiator device using MIDI-CI System Exclusive Messages. Some examples of how MIDI-CI might be initiated could include:

- A MIDI-CI capable device should initiate a MIDI-CI session after its power on and boot up procedure is complete.
- A MIDI-CI session should be triggered when the user selects the MIDI-CI Start button or similar function on a MIDI device. That device will take the role of Initiator.
- A MIDI-CI capable device should initiate a MIDI-CI session after it is connected to a USB Host and connection to the USB-MIDI driver has been completed.
- The Application Programming Interface (API) and/or Driver for MIDI services on a host computer should notify all applications on the host when a new MIDI device is connected and discovered. The API or applications should initiate a MIDI-CI session to auto-configure interoperability with the new device.

3.2 Selecting Initiator

When 2 devices shared a bidirectional connection to each other, either device may choose to act in the Initiation role by starting a MIDI-CI session. In some cases, both devices want to act as the Initiator.

This is not a problem for Bidirectional Negotiation for Single Direction Settings (specifically Profile Negotiation and Property Exchange).

However, when MIDI-CI uses a Bidirectional Negotiation for Bidirectional Settings (specifically Protocol Negotiation), there can be a conflict between the 2 devices simultaneously vying to manage the connection. MIDI-CI provides mechanisms and rules for resolving conflicts. Sometimes 2 devices in a MIDI-CI communication both want to act as Initiator.

3.2.1 Assigning Authority to be Initiator

The following rules allow devices to assign the Initiator role to one of 2 connected devices in a Bidirectional Negotiation for Bidirectional Settings. In this version of MIDI-CI, these rules apply only to Protocol Negotiation. These do not apply to Profile Configuration or Property Exchange. Apply the rules in the following order:

1. First Inquiry

The MIDI-CI device that first sends a MIDI-CI inquiry assumes the role of Initiator. The MIDI-CI device that receives the initial inquiry assumes the role of Responder.

2. Authority Level Overrides First Inquiry (See Section 3.2.2)

A device may optionally refuse to act as a Responder only if it has a higher Authority Level as reported in an Authority Level field of a message.

When a 1st MIDI-CI device takes the role of Initiator, a 2nd MIDI-CI device may refuse to act as a Responder if the 2nd MIDI-CI device has a higher Authority Level than the 1st device's Authority Level. Instead of sending a Responder reply message, the 2nd MIDI-CI device may send an initial inquiry message to claim Initiator role. That message uses the same MNID as was assigned by the 1st device. When the 1st device receives that inquiry in reply from the 2nd device using the same MNID as the 1st device's inquiry, it must change its role to Responder.

3. Simultaneous Inquiries = Use Authority Level (See Section 3.2.2)

In some cases, both devices may try to initiate a MIDI-CI session at the same time. In those cases, the devices shall use their Authority Level fields to determine which device continues as Initiator.

When a device sends an initial inquiry message it takes the role of Initiator. If the device receives an initial inquiry message at the same time as or following its own initial inquiry message, it shall compare its own Authority Level to the Authority Level of the incoming message. If the incoming message has a higher Authority Level, it shall set its role to Responder and send a reply message using the MNID of the message it received.

If the device's own Authority Level to the Authority Level of the incoming message are the same, the device will use the MNID to assign authority according to the following rule.

4. Same Authority Level = Use Highest MNID (See Section 3.2.3)

When 2 devices simultaneously try to take the Initiator role and both devices are of the same Authority Level, the device that has the highest value for MNID shall take the role of Initiator.

The device that sent the lower MNID shall set its role to Responder and send a reply message using the MNID of the message it received.

3.2.2 Authority Level

MIDI-CI provides an Authority Level, a 1 byte field with integer value, to designate devices that have management authority (the Initiator role) as compared to other devices. Devices are prioritized in the following order (highest integer value = highest level of authority):

0x70-0x7F Reserved

0x60-0x6F Highest Authority Level

0x50-0x5F

0x40-0x4F

0x30-0x3F

0x20-0x2F

0x10-0x1F Lowest Authority Level

0x00-0x0F Reserved

Descriptions and requirements of those devices follow. These are not strict requirements, sometimes it is difficult to classify a device to any defined type. These are just recommendations to help device manufacturers decide what value to assign to the device's Authority Level field.

While these are just recommendations, no devices should claim highest level of authority if they do not provide a wide range of MIDI services and API to manage multiple connected devices as are typically found in a general-purpose PC.

AUTHORITY LEVEL	RECOMMENDED FOR THESE TYPICAL DEVICES
0x70-0x7F	Reserved
0x60-0x6F Highest Authority Level	MIDI Node Server (PC)
0x50-0x5F	MIDI Gateway
0x40-0x4F	MIDI Translator
0x30-0x3F	MIDI Endpoint
0x20-2F	MIDI Event Processor (Sequencer, Arpeggiator)
0x10-1F Lowest Authority Level	MIDI Transport
0x00-0x0F	Reserved

3.2.3 MNID (MIDI Negotiation Identifier)

A 28bit random number ID generated by the initiator of the MIDI-CI Initial Inquiry and included (unchanged) in all subsequent MIDI-CI messages to help track the link between Initiator and Responder. Initiator and Responder use the same MNID in all MIDI-CI messages throughout all stages of a MIDI-CI Session. The same MNID may be used in subsequent MIDI-CI sessions between the same Initiator and Responder devices or a new MNID may be generated at the start of any new MIDI-CI Session. A new MNID is generated on any change of topology, or when Initiator is powered up.

3.2.4 User Selected Initiator

Some devices may allow a user to trigger a MIDI-CI inquiry from that device. In most cases, other devices will allow the device of user's choice to take authority.

> One typical example implementation might be a "MIDI-CI AutoConfiguration" button on the front panel of a device. This is useful for a controller keyboard connected to a DAW and several plugins. As the user changes plugins the "MIDI-CI AutoConfiguration" performs dynamic configuration changes on the controller keyboard to match the state or functions of the currently selected plugin.

The primary application for User Selected Initiator is to perform Profile Configuration and Property Exchange.

The device manufacturer should take care in implementation of this function to avoid triggering unnecessary system changes. In particular, this function may perform Protocol Negotiation once upon startup but it should not trigger Protocol Negotiation again until a power cycle or there has been a topology change that necessitates a new Protocol Negotiation.

3.3 MIDI-CI Session - Order of Processing

Devices do not need to support all categories of MIDI-CI implementation.

The first time a MIDI-CI Session is started the devices must proceed through Inquiry and Negotiation in the following order for any categories of MIDI-CI that they support.

1. Protocol Negotiation
2. Profile Configuration (includes Manufacturer and Model Information)
3. Property Exchange

If either the Initiator or Responder do not support any category of MIDI-CI Inquiry and Negotiation, then the MIDI-CI Session proceeds with the next category of MIDI-CI.

After this first set of Inquiries and Negotiations take place, a device can freely use any Inquiry or Negotiation message as required and independently from other categories or messages of Inquiry and Negotiation.

3.4 MIDI-CI Inquiry & Negotiation Messages

MIDI-CI Inquiries and Negotiations are accomplished using Universal System Exclusive messages. All messages conform to a common format with header and data as follows.

Standard Format for MIDI-CI Messages

All MIDI-CI messages use 0x0D as value for Universal System Exclusive Sub ID #1.

The Universal System Exclusive Sub ID #2 determines the category of message and the function of each message.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	DeviceID: Source or Destination (depending on message type) 7F = to/from MIDI Port 00-0F = to/from MIDI Channels 1-16 10-7E = Reserved
0D	Universal System Exclusive Sub ID #1: MIDI-CI
1 byte	Universal System Exclusive Sub ID #2: Type of MIDI-CI Message 0x00-0F Reserved 0x10-1F Protocol Negotiation Messages 0x20-2F Profile Configuration Messages 0x30-3F Property Exchange Messages 0x40-7E Reserved 0x7F NAK
1byte	MIDI-CI Message Version/Format
4 bytes	MNID (MIDI Negotiation Identifier)
nb bytes	Data
F7	End Universal System Exclusive

Source or Destination (depending on message type)

Values in this Universal System Exclusive DeviceID are used for Channel addressing. Values 0x00-0xF map to MIDI Channels 1-16. Messages with a Value of 0x7F are addressed to the whole MIDI port/cable, not channelized.

Some MIDI-CI messages, such as Protocol Negotiation messages, are only valid when sent to a whole MIDI Port 0x7F. Protocol Negotiation messages are only valid for a whole MIDI Port. Profile Configuration and Property Exchange messages can address a whole MIDI Port or individual Channels on a MIDI Port. See individual messages as defined in this document for whether each message can be used on a per channel basis.

In a MIDI-CI inquiry message sent by the Initiator, this is the destination of the inquiry. Default value is 0x7F = MIDI Port based inquiry. But value can be 0x00-0x0F to address a specific MIDI Channel.

In a MIDI-CI reply message sent by the Responder, this is the source of the reply. Default value is 0x7F = MIDI Port based reply. But the value can be 0x00-0x0F to reply about a specific MIDI Channel.

Values 0x00-0x0F are for 16 MIDI channels. This allows inquiries and negotiation on a specific channel. This is useful for using Profile Configuration and Property Exchange on a per channel basis. This cannot be used for Protocol Negotiation.

1 byte MIDI-CI Message Version/Format

In this 1.0 version of the MIDI-CI specification, the version number is 0x00

4 bytes MNID (MIDI Negotiation Identifier)

A 28bit random number ID generated by the initiator of the MIDI-CI Initial Inquiry and included (unchanged) in all subsequent MIDI-CI messages to help track the link between Initiator and Responder. Initiator and Responder use the same MNID in all MIDI-CI messages throughout all stages of a MIDI-CI Session. The same MNID may be used in subsequent MIDI-CI sessions between the same Initiator and Responder devices or a new MNID may be generated at the start of any new MIDI-CI Session. A new MNID is generated on any change of topology, or when Initiator is powered up.

Data

If the message contains any payload it is in this field.

Some MIDI-CI messages define that the first 11 bytes of the Data section identify the device as follows:

3bytes	Device Manufacturer (System Exclusive ID Number)
2bytes	Device Family
2bytes	Device Family Model Number
4bytes	Device Revision Level

3 bytes Device Manufacturer

This is the System Exclusive ID of the device manufacturer. For System Exclusive ID values that are only 1 byte in length, the System Exclusive ID value is in the first byte and the remaining 2 bytes are filled with zeroes: ID 00 00

2 bytes Device Family

This identifies the related group of models to which the device belongs. The manufacturer is free to determine the grouping of models and the format of the data in this field.

2 bytes Device Family Model Number

This identifies a specific model from the Device Manufacturer. The manufacturer is free to determine the assignment of values and the format of the data in this field.

4 bytes Device Revision Level

This is the version number of a device model number. This is typically version of software or firmware but may also be version of hardware. If a model undergoes any version update or other design change that changes its midi implementation or capabilities as may be discovered by MIDI-CI (including Property Exchange), then this Device Revision Level must be changed. The manufacturer is free to determine the format of the data in this field.

3.5 MIDI-CI Messages Format and Protocols

MIDI-CI enables switching between various protocol types. This MIDI-CI specification defines MIDI-CI messages using a MIDI Universal System Exclusive message for use with any protocol that supports System Exclusive.

If MIDI-CI is used to negotiate to an AMEI / MMA standard protocol that does not support System Exclusive, that protocol must define equivalent MIDI-CI messages using native messages of that protocol.

If MIDI-CI is used to negotiate to a Protocol Set by SysEx ID (manufacturer specific protocol), the manufacturer is free to decide whether to define equivalent MIDI-CI messages using native messages of that protocol. MIDI-CI recommends that the manufacturer should add native messages of that protocol for negotiating back to AMEI / MMA standard protocols.

3.6 MIDI-CI Proxy Device

MIDI Node Servers, MIDI Gateways, and MIDI Translators manage connections and pass MIDI streams to different ports or via different protocols. Example devices include a MIDI API in a general-purpose computer, Plugin API, MIDI-DIN to USB-MIDI converters, MIDI processors (hardware or in software), DAWs with multiple Plugins, or networked MIDI bridges like RTP-MIDI. There are multiple options for if and how MIDI Node Servers, MIDI Gateways, and MIDI Translators may engage in the MIDI-CI connection.

1. If a MIDI Node Server, MIDI Gateway, or MIDI Translator merely passes the MIDI stream through, the general recommendation is to act as a transparent bridge and pass on the MIDI-CI messages without modification. MIDI-CI enabled devices on both ends of the MIDI Node Server, MIDI Gateway, or MIDI Translator will be able to establish a session directly.

2. Sometimes a MIDI Node Server, MIDI Gateway, or MIDI Translator may act as a **MIDI-CI Proxy** for other devices. It reports the manufacturer and other fields with values from the other device and not its own. It is important, however, to not misrepresent the device: fields should only be populated with known values (e.g. from USB descriptors or Plugin API properties). Do not guess values or use arbitrary default values. Some example applications include:

   - If a MIDI Node Server or MIDI Gateway is managing routing between various devices, it may need to inform connected devices of routing changes. The MIDI Node Server or MIDI Gateway may act as a MIDI-CI Proxy to inform devices of Profile changes triggered by connection or topology changes.
   - If a device that is NOT capable of MIDI-CI is connected to a MIDI Node Server, MIDI Gateway, or MIDI Translator, and if the MIDI Node Server, MIDI Gateway, or MIDI Translator has some knowledge about that device, it is recommended that the MIDI Node Server, MIDI Gateway, or MIDI Translator act as a MIDI-CI Proxy: it engages in MIDI-CI negotiation in place of that device.

3. In some cases, it makes sense for a MIDI Node Server, MIDI Gateway, or MIDI Translator to engage in the MIDI-CI negotiation as itself. For example, if it modifies the MIDI data stream in a way that is incompatible with MIDI-CI such as converting to a non-MIDI protocol, or merging/duplicating MIDI streams.

In all cases, it is up to the manufacturer to decide the most sensible approach. MIDI Node Servers, MIDI Gateways, and MIDI Translators which can be configured by the user can offer different options to be selected by the user.

3.7 MIDI-CI NAK Message

The MIDI-CI NAK message is used to respond to any message that a device does not understand. Examples of application for this NAK message include:

- Reply to a MIDI-CI message the device does not support
- Reply to a MIDI-CI message with MIDI-CI Message Version/Format the device does not support*
- Reply to a malformed MIDI-CI message

Receiver response to a NAK message is undefined.

* Note: Intended Goal and Recommendation for Future MIDI-CI revisions:

  When a future MIDI-CI Version >1.0 Device#1 receives a MIDI-CI 1.0 NAK from a Device#2, the Device#1 will know that the Device#2 only supports MIDI-CI version 1.0. The action taken by the Device#1 is not defined in this specification but may be defined in a future MIDI-CI Version >1.0 specification.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	DeviceID: Source or Destination (depending on message type) 7F = to/from MIDI Port 00-0F = to/from MIDI Channels 1-16 10-7E = Reserved
0D	Universal System Exclusive Sub ID #1: MIDI-CI
7F	MIDI-CI NAK
1byte	MIDI-CI Message Version/Format
4 bytes	MNID (MIDI Negotiation Identifier)
F7	End Universal System Exclusive

4. PROTOCOL NEGOTIATION

This mechanism selects a new protocol for communication between MIDI-CI devices. The Protocol used between MIDI-CI devices is common to both directions of a bidirectional connection. If Protocol used is changed in one direction, the Protocol in the opposite direction is changed at the same time.

4.1 Protocol Types Supported

MIDI-CI enables switching to protocols supported by 2 devices connected to each other. The choice of protocols available in this revision of MIDI-CI are:

0x00 Protocols Set by System Exclusive ID
0x01 MIDI 1.0
0x02 New AMEI/MMA Protocol*

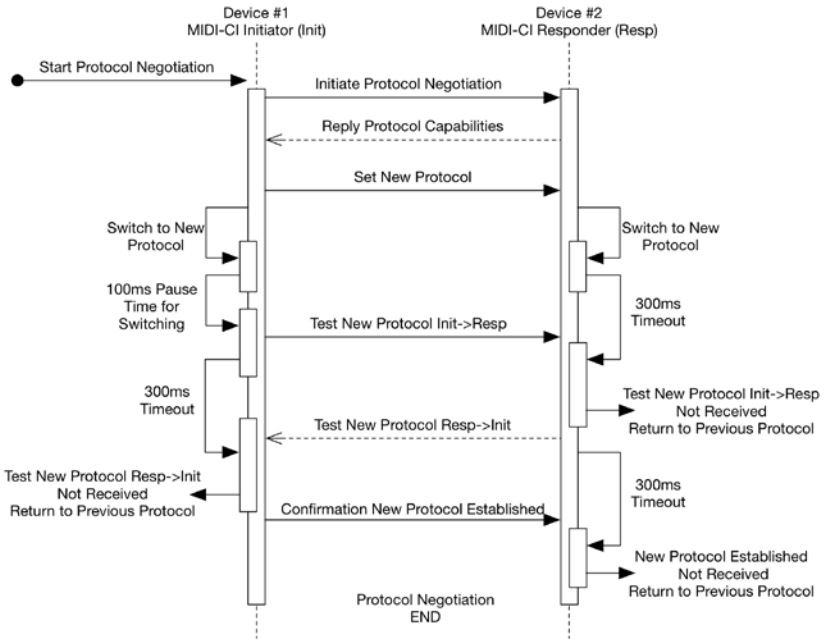
Values 0x03-0x7F are Reserved

**There is a new MIDI protocol standard under development within AMEI/MMA. That new protocol will be assigned the 0x02 value. After the protocol is finalized, an updated MIDI-CI specification will show the name of the protocol anywhere that this document says "New AMEI/MMA Protocol".*

4.2 Protocol Inquiry and Negotiation Mechanism

Initiator begins Protocol Negotiation with Initiate Protocol Negotiation. When Responder is ready to switch Protocols, it replies with Responder Reply Protocol Capabilities Message. At this point Initiator can decide a Timeout before escaping negotiation or restarting negotiation. Other Timeout and escapes are defined and shown in the diagram below.

In case of failure, both devices always return to "Previous Protocol". In many cases Previous Protocol will be MIDI 1.0. But in some cases, devices might be using some other Protocol (Example New AMEI/MMA Protocol) and then start a negotiation to yet another Protocol (Example Manufacturer Specific). If that negotiation fails, the return to Previous Protocol is not MIDI 1.0 (Example New AMEI/MMA Protocol).



4.3 Initiate Protocol Negotiation Message

This initial message is both an Inquiry and a Report of Initiator capabilities.

<table>
<tr><th>Value</th><th>Parameter</th></tr>
<tr><td>F0</td><td>System Exclusive Start</td></tr>
<tr><td>7E</td><td>Universal System Exclusive</td></tr>
<tr><td>7F</td><td>To/From whole MIDI Port</td></tr>
<tr><td>0D</td><td>Universal System Exclusive Sub ID #1: MIDI-CI</td></tr>
<tr><td>10</td><td>Universal System Exclusive Sub ID #2: Initiate Protocol Negotiation</td></tr>
<tr><td>00</td><td>MIDI-CI Message Version/Format</td></tr>
<tr><td>4 bytes</td><td>MNID (MIDI Negotiation Identifier)</td></tr>
<tr><td>1byte</td><td>Authority Level</td></tr>
<tr><td>3bytes</td><td>Device Manufacturer (System Exclusive ID Number)</td></tr>
<tr><td>2bytes</td><td>Device Family</td></tr>
<tr><td>2bytes</td><td>Device Family Model Number</td></tr>
<tr><td>4bytes</td><td>Device Revision Level</td></tr>
<tr><td>1 byte</td><td>Number of Supported Protocols (np)</td></tr>
<tr><td>5 bytes</td><td>Preferred Protocol Type:
<table>
<tr><td>Protocol Byte 1</td><td>0x00= Protocol Set by SysEx ID</td><td>0x01=MIDI 1.0</td><td>0x02=New AMEI/MMA Protocol</td></tr>
<tr><td>Protocol Byte 2</td><td>Manufacturer ID Byte#1</td><td>Reserved *1
Set to 0x00</td><td>Version</td></tr>
<tr><td>Protocol Byte 3</td><td>Manufacturer ID Byte#2</td><td>Reserved *1
Set to 0x00</td><td>SubType</td></tr>
<tr><td>Protocol Byte 4</td><td>Manufacturer ID Byte#3</td><td>Reserved *1
Set to 0x00</td><td>Reserved *1
Set to 0x00</td></tr>
<tr><td>Protocol Byte 5</td><td>Type</td><td>Reserved *1
Set to 0x00</td><td>Reserved *1
Set to 0x00</td></tr>
</table>
*1: Reserved field value is 0x00 (null). Other values may be defined in future specifications.</td></tr>
<tr><td rowspan="3">(np-1)x5 bytes</td><td>Optional: Another Supported Protocol in 5 bytes.</td></tr>
<tr><td>…</td></tr>
<tr><td>Optional: Last Supported Protocol</td></tr>
<tr><td>F7</td><td>End Universal System Exclusive</td></tr>
</table>

1 byte Authority Level

MIDI-CI provides an Authority Level, a 1 byte field with integer value, to designate devices that have management authority (the Initiator role) as compared to other devices. Devices are prioritized in descending order. Larger integer value = higher level of authority.

0x70-0x7F Reserved

0x60-0x6F Highest Authority Level

0x50-0x5F

0x40-0x4F

0x30-0x3F

0x20-0x2F

0x10-0x1F Lowest Authority Level

0x00-0x0F Reserved

Supported Protocols:

Each Protocol supported by a Device is listed with a set of 5 bytes (Protoc Byte 1 - Protoc Byte 5). The Protocol that is preferred by the device should be the first in the list of Supported Protocols.

All Devices must support MIDI 1.0 as one of the Protocol choices. If a device only supports one Protocol, it must be MIDI 1.0.

The preferred choice of Protocol for the device should be the first one listed. Other Protocols supported should be listed in order of preference.

Sub Type: (New AMEI/MMA Protocol only)

Sub Type differentiates between variants of a Protocol. The values of SubTypes are to be defined by that Protocol specification. For Example, New AMEI/MMA Protocol SubTypes might include:

0x00 = Extended MIDI Messages

0x01 = Extended MIDI Messages with 32bit Timestamps

Some SubTypes may be a superset of other SubTypes. If a device supports such a superset SubType and also supports the SubType which is a subset of that superset, then the device must declare both SubTypes in the list of supported protocols.

Protocol Set by SysEx ID, Type:

Manufacturers can use MIDI-CI to negotiate to a new Protocol of their own proprietary design by using their own System Exclusive ID. Each Manufacturer SysEx ID can have up to 128 different protocol types. Manufacturer is free to determine their own method for negotiating back from their own protocol to the standard MIDI Protocols.

4.4 Reply Protocol Capabilities Message

<table>
<tr><th>Value</th><th>Parameter</th></tr>
<tr><td>F0</td><td>System Exclusive Start</td></tr>
<tr><td>7E</td><td>Universal System Exclusive</td></tr>
<tr><td>7F</td><td>To/From whole MIDI Port</td></tr>
<tr><td>0D</td><td>Universal System Exclusive Sub ID #1: MIDI-CI</td></tr>
<tr><td>11</td><td>Universal System Exclusive Sub ID #2: Responder Reply Protocol Capabilities</td></tr>
<tr><td>00</td><td>MIDI-CI Message Version/Format</td></tr>
<tr><td>4 bytes</td><td>MNID (MIDI Negotiation Identifier)</td></tr>
<tr><td>1byte</td><td>Authority Level</td></tr>
<tr><td>3bytes</td><td>Device Manufacturer (System Exclusive ID Number)</td></tr>
<tr><td>2bytes</td><td>Device Family</td></tr>
<tr><td>2bytes</td><td>Device Family Model Number</td></tr>
<tr><td>4bytes</td><td>Device Revision Level</td></tr>
<tr><td>1 byte</td><td>Number of Supported Protocols (np)</td></tr>
<tr><td>5bytes</td><td>Preferred Protocol Type:
<table>
<tr><td>Protocol Byte 1</td><td>0x00= Protocol Set by SysEx ID</td><td>0x01= MIDI 1.0</td><td>0x02= New AMEI/MMA Protocol</td></tr>
<tr><td>Protocol Byte 2</td><td>Manufacturer ID Byte#1</td><td>Reserved *1 Set to 0x00</td><td>Version</td></tr>
<tr><td>Protocol Byte 3</td><td>Manufacturer ID Byte#2</td><td>Reserved *1 Set to 0x00</td><td>SubType</td></tr>
<tr><td>Protocol Byte 4</td><td>Manufacturer ID Byte#3</td><td>Reserved *1 Set to 0x00</td><td>Reserved *1 Set to 0x00</td></tr>
<tr><td>Protocol Byte 5</td><td>Type</td><td>Reserved *1 Set to 0x00</td><td>Reserved *1 Set to 0x00</td></tr>
</table>
*1: Reserved field value is 0x00 (null). Other values may be defined in future specifications.</td></tr>
<tr><td rowspan="3">(np-1)x5 bytes</td><td>Optional: Another Supported Protocol in 5 bytes.</td></tr>
<tr><td>...</td></tr>
<tr><td>Optional: Last Supported Protocol</td></tr>
<tr><td>F7</td><td>End Universal System Exclusive</td></tr>
</table>

Supported Protocols:

The Responder usually replies with a list of all Protocols that it can support, in order of preference of the Responder. However, the Responder may optionally adapt its list of supported Protocols to leave out protocols that are not supported by the Initiator (as reported in the Initiate Protocol Negotiation Message).

4.5 Set New Protocol Message

The Initiator selects new Protocol based on matching its own capabilities against the capabilities of the Responder. Initiator sends the newly selected Protocol to the Responder.

<table>
<tr><th>Value</th><th>Parameter</th></tr>
<tr><td>F0</td><td>System Exclusive Start</td></tr>
<tr><td>7E</td><td>Universal System Exclusive</td></tr>
<tr><td>7F</td><td>To/From whole MIDI Port</td></tr>
<tr><td>0D</td><td>Universal System Exclusive Sub ID #1: MIDI-CI</td></tr>
<tr><td>12</td><td>Universal System Exclusive Sub ID #2: Set New Selected Protocol</td></tr>
<tr><td>00</td><td>MIDI-CI Message Version/Format</td></tr>
<tr><td>4 bytes</td><td>MNID (MIDI Negotiation Identifier)</td></tr>
<tr><td>1byte</td><td>Authority Level</td></tr>
<tr><td>5 bytes</td><td>New Protocol Type:
<table>
<tr><td>Protocol Byte 1</td><td>0x00= Protocol Set by SysEx ID</td><td>0x01= MIDI 1.0</td><td>0x02= New AMEI/MMA Protocol</td></tr>
<tr><td>Protocol Byte 2</td><td>Manufacturer ID Byte#1</td><td>Reserved *1 Set to 0x00</td><td>Version</td></tr>
<tr><td>Protocol Byte 3</td><td>Manufacturer ID Byte#2</td><td>Reserved *1 Set to 0x00</td><td>SubType</td></tr>
<tr><td>Protocol Byte 4</td><td>Manufacturer ID Byte#3</td><td>Reserved *1 Set to 0x00</td><td>Reserved *1 Set to 0x00</td></tr>
<tr><td>Protocol Byte 5</td><td>Type</td><td>Reserved *1 Set to 0x00</td><td>Reserved *1 Set to 0x00</td></tr>
</table>
*1: Reserved field value is 0x00 (null). Other values may be defined in future specifications.</td></tr>
<tr><td>F7</td><td>End Universal System Exclusive</td></tr>
</table>

After the Initiator sends this Set New Protocol message, it switches its own Protocol while also waiting 100ms to allow the Responder to switch Protocol. The Initiator then sends the next message, Test New Protocol Initiator to Responder. (100ms is a guideline)

After the Responder receives this Set New Protocol message, it switches its own Protocol. It also starts a 300ms timeout with expectation of receiving the Test New Protocol Initiator to Responder message from the Initiator. (300ms is a guideline)

4.6 Test New Protocol Initiator to Responder Message

The Initiator sends this confirmation test message that Responder uses to confirm that the Protocol has been successfully established between Initiator MIDI Out and Responder MIDI In.

This test message is defined using a MIDI System Exclusive message for use with any protocol that supports System Exclusive. If MIDI-CI is used to negotiate to a protocol that does not support System Exclusive, that protocol must define a similar test using native messages of that protocol.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
7F	To/From whole MIDI Port
0D	Universal System Exclusive Sub ID #1: MIDI-CI
13	Universal System Exclusive Sub ID #2: Test New Protocol Initiator to Responder
00	MIDI-CI Message Version/Format
4 bytes	MNID (MIDI Negotiation Identifier)
1byte	Authority Level
48 bytes	Test Data: string of 48 numbers in ascending order: 0x00, 0x01, 0x02 ... 0x2E, 0x2F.
F7	End Universal System Exclusive

After the Initiator sends this test message, it starts a 300ms timeout counter with expectation of receiving a Test New Protocol Responder to Initiator message from the Responder. (300ms is a guideline)

After the Responder successfully receives this test message, it replies with the next message, the Test New Protocol Responder to Initiator.

If the Responder does not successfully receive this test message before its 300ms timeout counter expires, it resets its Protocol to the previous value. (300ms is a guideline)

4.7 Test New Protocol Responder to Initiator Message

The Responder sends this confirmation test message that Initiator uses to confirm that the Protocol has been successfully established between Initiator MIDI Out and Responder MIDI In (via previous test) and between Responder MIDI Out and Initiator MIDI In.

This test message is defined using a MIDI System Exclusive message for use with any protocol that supports System Exclusive. If MIDI-CI is used to negotiate to a protocol that does not support System Exclusive, that protocol must define a similar test using native messages of that protocol.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
7F	To/From whole MIDI Port
0D	Universal System Exclusive Sub ID #1: MIDI-CI
14	Universal System Exclusive Sub ID #2: Test New Protocol Responder to Initiator
00	MIDI-CI Message Version/Format
4 bytes	MNID (MIDI Negotiation Identifier)
1byte	Authority Level
48 bytes	Test Data: string of 48 numbers in ascending order: 0x00, 0x01, 0x02 ... 0x2E, 0x2F.
F7	End Universal System Exclusive

After the Initiator successfully receives this test message, it replies with the next message, the Confirmation New Protocol Established.

If the Initiator does not successfully receive this test message before its 300ms timeout counter expires, it resets its Protocol to the previous value. Then the Initiator can decide whether to restart Protocol Negotiation. (300ms is a guideline)

4.8 Confirmation New Protocol Established Message

The Initiator sends this confirmation test message that Responder uses to confirm that the Protocol has been successfully established.

This confirmation message is defined using a MIDI System Exclusive message for use with any protocol that supports System Exclusive. If MIDI-CI is used to negotiate to a protocol that does not support System Exclusive, that protocol must define a similar confirmation using native messages of that protocol.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
7F	To/From whole MIDI Port
0D	Universal System Exclusive Sub ID #1: MIDI-CI

15	Universal System Exclusive Sub ID #2: Confirmation New Protocol Established
00	MIDI-CI Message Version/Format
4 bytes	MNID (MIDI Negotiation Identifier)
1byte	Authority Level
F7	End Universal System Exclusive

Messages can now be exchanged between the 2 devices using the newly established Protocol.

5. PROFILE CONFIGURATION

Profiles define specific implementations of a set of MIDI messages chosen to suit a particular instrument, device type, or to accomplish a particular task. Two devices that conform to the same Profile will have generally have greater interoperability between them than devices using MIDI without Profiles. Profiles increase interoperability and ease of use while lowering the need for manual configuration of devices by users.

5.1 Profile Configuration Mechanism

Profiles are controlled by the following Common Profile Configuration Messages:

- Initiator Profile Inquiry
- Responder Reply Profile Capabilities
- Set Profile On
- Set Profile Off
- Profile Enabled Report
- Profile Disabled Report

More information about Profiles is defined in other specifications of MMA and AMEI.

5.2 Profile Inquiry Message

A device (Initiator) sends this to request a list of Profiles that a connected Responder device supports.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Destination 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub ID #1: MIDI-CI
20	Universal System Exclusive Sub ID #2: Profile Inquiry
00	MIDI-CI Message Version/Format
4 bytes	MNID (MIDI Negotiation Identifier)
3bytes	Manufacturer (System Exclusive ID Number)
2bytes	Device Family
2bytes	Device Family Model Number
4bytes	Device Revision Level
F7	End Universal System Exclusive

5.3 Responder Reply Profile Capabilities Message

When a device (Responder) receives the Profile Inquiry Message it may reply with this message to report a list of Profiles the device (Responder) supports.

There are 2 lists of Supported Profiles in the message:

1. Profiles that are Supported and Currently Enabled
2. Profiles that are Supported but Currently Disabled

The Initiator may use this information to auto-configure the connection between the devices for increased interoperability.

<table>
<tr><th>Value</th><th>Parameter</th></tr>
<tr><td>F0</td><td>System Exclusive Start</td></tr>
<tr><td>7E</td><td>Universal System Exclusive</td></tr>
<tr><td>1 byte</td><td>Source
7F = to/from whole MIDI Port
00-0F = to/from MIDI Channels 1-16</td></tr>
<tr><td>0D</td><td>Universal System Exclusive Sub ID #1: MIDI-CI</td></tr>
<tr><td>21</td><td>Universal System Exclusive Sub ID #2: Responder Reply Profile Capabilities</td></tr>
<tr><td>00</td><td>MIDI-CI Message Version/Format</td></tr>
<tr><td>4 bytes</td><td>MNID (MIDI Negotiation Identifier)</td></tr>
<tr><td>3bytes</td><td>Device Manufacturer (System Exclusive ID Number)</td></tr>
<tr><td>2bytes</td><td>Device Family</td></tr>
<tr><td>2bytes</td><td>Device Family Model Number</td></tr>
<tr><td>4bytes</td><td>Device Revision Level</td></tr>
<tr><td>2bytes</td><td>Number of Currently-Enabled Profiles (cep)</td></tr>
<tr><td>5 bytes</td><td>Profile ID of First Currently-Enabled Profile:
<table>
<tr><td>Profile ID Byte 1</td><td>0x7E Standard Defined Profile</td><td>Manufacturer SysEx ID 1 Profile</td></tr>
<tr><td>Profile ID Byte 2</td><td>Profile Number MSB</td><td>Manufacturer SysEx ID 2 Profile</td></tr>
<tr><td>Profile ID Byte 3</td><td>Profile Number LSB</td><td>Manufacturer SysEx ID 3 Profile</td></tr>
<tr><td>Profile ID Byte 4</td><td>Profile Version</td><td>Manufacturer Specific Info</td></tr>
</table></td></tr>
</table>

	<table><tr><td>Profile ID Byte 5</td><td>Profile Level</td><td>Manufacturer Specific Info</td></tr></table>
(cep - 1) x 5 bytes	Optional: Profile ID of Other Currently-Enabled Profiles in sets of 5 bytes. ... Optional: Profile ID of Last Currently-Enabled Profile
2bytes	Number of Currently-Disabled Profiles Supported (cdp)
5bytes	Profile ID of First Currently-Disabled Profile Supported
(cdp - 1) x 5 bytes	Optional: Profile ID of Other Currently-Disabled Profiles Supported in sets of 5 bytes. ... Optional: Profile ID of Last Currently-Disabled Profile Supported
F7	End Universal System Exclusive

Profile ID

Each Profile has a 5 byte identifier. Standard Defined Profiles are those adopted by AMEI and MMA. Each one uses the ID defined by each Profile Specification and by other AMEI/MMA Profile related specifications. The value of the Profile ID Byte 1 is 0x7E (Universal).

Manufacturers may use MIDI-CI to control Profiles of their own proprietary design by using their own System Exclusive ID. Each Manufacturer SysEx ID can freely use the 2 bytes of Manufacturer Specific Info. These 2 bytes allow up to 16384 different Manufacturer Profile Numbers.

Profile ID Byte 1	0x7E Standard Defined Profile	Manufacturer SysEx ID 1 Profile
Profile ID Byte 2	Profile Number MSB	Manufacturer SysEx ID 2 Profile
Profile ID Byte 3	Profile Number LSB	Manufacturer SysEx ID 3 Profile
Profile ID Byte 4	Profile Version	Manufacturer Specific Info
Profile ID Byte 5	Profile Level	Manufacturer Specific Info

Currently Enabled

These are Profiles that the device supports and that are currently active at the time of inquiry. Some devices may have a Profile that is already active (Enabled) before receiving a Set Profile On Message. For example, a MIDI acoustic piano might be fixed to conform to a Piano Profile Specification; Piano Profile is always Enabled.

Currently Disabled

These are Profiles that a device can support but that are not currently active. When a Profile on a device is not active (Disabled), the device does not currently conform to the requirements of the Profile specification. But the device can be switched to conform to the requirements of the Profile specification using the Profile Enable On message.

5.4 Set Profile On Message

A transmitter sends this to enable a Profile on a receiving device. (May be sent by Initiator or Responder.)

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Destination 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub ID #1: MIDI-CI
22	Universal System Exclusive Sub ID #2: Set Profile On
00	MIDI-CI Message Version/Format
4 bytes	MNID (MIDI Negotiation Identifier)
5 bytes	Profile ID of Profile to be Set to On (to be Enabled)
F7	End Universal System Exclusive

5.5 Set Profile Off Message

A transmitter sends this to disable a Profile on a receiving device. (May be sent by Initiator or Responder.)

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive

1 byte	Destination 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub ID #1: MIDI-CI
23	Universal System Exclusive Sub ID #2: Set Profile Off
00	MIDI-CI Message Version/Format
4 bytes	MNID (MIDI Negotiation Identifier)
5 bytes	Profile ID of Profile to be Set to Off (to be Disabled)
F7	End Universal System Exclusive

5.6 Profile Enabled Report Message

A device sends this message if it has enabled a Profile.
This is an acknowledgement upon receipt of a Set Profile On message.
This is an informative message if any other event enables a Profile.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Source 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub ID #1: MIDI-CI
24	Universal System Exclusive Sub ID #2: Profile Enabled
00	MIDI-CI Message Version/Format
4 bytes	MNID (MIDI Negotiation Identifier)
5bytes	Profile ID of Profile that is now Enabled
F7	End Universal System Exclusive

5.7 Profile Disabled Report Message

A device sends this message if it has disabled a Profile.
This is an acknowledgement upon receipt of a Set Profile Off message.
This is an informative message if any other event disables a Profile.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Source 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub ID #1: MIDI-CI
25	Universal System Exclusive Sub ID #2: Profile Disabled
00	MIDI-CI Message Version/Format
4 bytes	MNID (MIDI Negotiation Identifier)
5 bytes	Profile ID of Profile that is now Disabled
F7	End Universal System Exclusive

6. PROPERTY EXCHANGE

Property Exchange is used to Inquire, Get, and Set many properties including but not limited to device configuration settings, a list of controllers and resolution, a list of patches with names and other meta data, manufacturer, model number, and version.
Detailed information about the Property Data used in Property Exchange messages is defined in another specification. The MIDI-CI specification contains only the base messages used for Property Exchange. Most of the definition for implementing Property Exchange, including all defined properties and semantics, exists outside of the MIDI-CI specification.

6.1 Property Inquiry and Negotiation Mechanism

Properties are exchanged by the following Common Property Exchange Messages:

- Inquiry Property Exchange Capabilities
- Reply to Property Exchange Capabilities
- Inquiry Has Property
- Reply to Has Property
- Inquiry Get Property
- Reply to Get Property
- Inquiry Set Property
- Reply to Set Property
- Notify Message
- NAK

See also “6.13 General Mechanisms of Property Exchange”.

6.2 Data Sets May Be Sent in Multiple Chunks

A device may choose to send a Data Set of a Property Exchange as a single SysEx message or as a set of multiple SysEx messages or “Chunks”.
If the device chooses to send a Data Set in multiple Chunks, it specifies the “Number of Chunks in Data Set” and labels each Chunk with a sequential “Number of This Chunk”. The Number of This Chunk always starts counting from a value of 0x0001.
If the sender device does not know the total number of Chunks in advance, the device sets the Number of Chunks in Data Set to 0x0000. Then when sending Chunks, the sender sets the Number of This Chunk for each Chunk in a sequential count as usual. However, when Number of Chunks in Data Set is unknown (0x0000) the Number of This Chunk for the final Chunk must be set to 0x0000.
If a sender terminates a Data Set before sending the correct number of Chunks to match Number of Chunks in Data Set, then the final chunk it sends should be labeled with Number of This Chunk = 0x0000
The Data Set always ends upon either of the following:

1. Number of This Chunk = Number of Chunks in Data Set
2. Number of This Chunk = 0x0000

6.3 Inquiry Property Exchange Capabilities

A device (Initiator) sends this to request basic information for sending and receiving subsequent Property Exchange messages.

This inquiry does not need to be performed for every Property Exchange session.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Destination 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub ID #1: MIDI-CI
30	Universal System Exclusive Sub ID #2: Inquiry Property Exchange Capabilities
00	MIDI-CI Message Version/Format
4 bytes	MNID (MIDI Negotiation Identifier)
4 bytes	Receivable Maximum SysEx Message Size
F7	End Universal System Exclusive

6.4 Reply to Property Exchange Capabilities

When a device (Responder) receives the Inquiry Property Exchange Capabilities message it may reply with this message to report basic information for sending and receiving.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Source 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub ID #1: MIDI-CI
31	Universal System Exclusive Sub ID #2: Reply to Property Exchange Capabilities
00	MIDI-CI Message Version/Format
4 bytes	MNID (MIDI Negotiation Identifier)

4 bytes	Receivable Maximum SysEx Message Size
3bytes	Device Manufacturer (System Exclusive ID Number)
2bytes	Device Family
2bytes	Device Family Model Number
4bytes	Device Revision Level
F7	End Universal System Exclusive

6.5 Inquiry Has Property

A device (Initiator) sends this to discover if a receiving device (Responder) supports a Property and whether that Property is settable by a Property Exchange Inquiry Set Property message.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Destination 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub ID #1: MIDI-CI
32	Universal System Exclusive Sub ID #2: Inquiry Has Property
00	MIDI-CI Message Version/Format
4 bytes	MNID (MIDI Negotiation Identifier)
2 bytes	Number of Chunks in Data Set 0x0000 = Number of Chunks in Data Set is Unknown
2 bytes	Number of This Chunk (count starts from 0x0001) 0x0000 = Final Chunk when Number of Chunks in Data Set is Unknown
4 bytes	Length of Following Property Data
nn bytes	Property Data
F7	End Universal System Exclusive

6.6 Reply to Has Property

A device (Responder) sends this after receiving an Inquiry Has Property message.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Source 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub ID #1: MIDI-CI
33	Universal System Exclusive Sub ID #2: Reply to Has Property
00	MIDI-CI Message Version/Format
4 bytes	MNID (MIDI Negotiation Identifier)
2 bytes	Number of Chunks in Data Set 0x0000 = Number of Chunks in Data Set is Unknown
2 bytes	Number of This Chunk (count starts from 0x0001) 0x0000 = Final Chunk when Number of Chunks in Data Set is Unknown
4 bytes	Length of Following Property Data
nn bytes	Property Data
F7	End Universal System Exclusive

6.7 Inquiry Get Property

A device (Initiator) sends this to discover the value of a Property in a receiving device (Responder).

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Destination 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub ID #1: MIDI-CI
34	Universal System Exclusive Sub ID #2: Inquiry Get Property

00	MIDI-CI Message Version/Format
4 bytes	MNID (MIDI Negotiation Identifier)
2 bytes	Number of Chunks in Data Set 0x0000 = Number of Chunks in Data Set is Unknown
2 bytes	Number of This Chunk (count starts from 0x0001) 0x0000 = Final Chunk when Number of Chunks in Data Set is Unknown
4 bytes	Length of Following Property Data
nn bytes	Property Data
F7	End Universal System Exclusive

6.8 Reply to Get Property

A device (Responder) sends this after receiving an Inquiry Get Property message.

Value	**Parameter**
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Source 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub ID #1: MIDI-CI
35	Universal System Exclusive Sub ID #2: Reply to Get Property
00	MIDI-CI Message Version/Format
4 bytes	MNID (MIDI Negotiation Identifier)
2 bytes	Number of Chunks in Data Set 0x0000 = Number of Chunks in Data Set is Unknown
2 bytes	Number of This Chunk (count starts from 0x0001) 0x0000 = Final Chunk when Number of Chunks in Data Set is Unknown
4 bytes	Length of Following Property Data
nn bytes	Property Data
F7	End Universal System Exclusive

6.9 Inquiry Set Property

A device (Initiator) sends this to set the value of a Property in a receiving device (Responder).

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Destination 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub ID #1: MIDI-CI
36	Universal System Exclusive Sub ID #2: Inquiry Set Property
00	MIDI-CI Message Version/Format
4 bytes	MNID (MIDI Negotiation Identifier)
2 bytes	Number of Chunks in Data Set 0x0000 = Number of Chunks in Data Set is Unknown
2 bytes	Number of This Chunk (count starts from 0x0001) 0x0000 = Final Chunk when Number of Chunks in Data Set is Unknown
4 bytes	Length of Following Property Data
nn bytes	Property Data
F7	End Universal System Exclusive

6.10 Reply to Set Property

A device (Responder) sends this after receiving an Inquiry Set Property message.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Source 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub ID #1: MIDI-CI
37	Universal System Exclusive Sub ID #2: Reply to Set Property
00	MIDI-CI Message Version/Format

4 bytes	MNID (MIDI Negotiation Identifier)
2 bytes	Number of Chunks in Data Set 0x0000 = Number of Chunks in Data Set is Unknown
2 bytes	Number of This Chunk (count starts from 0x0001) 0x0000 = Final Chunk when Number of Chunks in Data Set is Unknown
4 bytes	Length of Following Property Data
nn bytes	Property Data
F7	End Universal System Exclusive

6.11 Notify Message

This is an informative message to report error messages, change of state, or other information.

Value	**Parameter**
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Source 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub ID #1: MIDI-CI
3F	Universal System Exclusive Sub ID #2: Notify Message
00	MIDI-CI Message Version/Format
4 bytes	MNID (MIDI Negotiation Identifier)
2 bytes	Number of Chunks in Data Set 0x0000 = Number of Chunks in Data Set is Unknown
2 bytes	Number of This Chunk (count starts from 0x0001) 0x0000 = Final Chunk when Number of Chunks in Data Set is Unknown
4 bytes	Length of Following Property Data
nn bytes	Property Data
F7	End Universal System Exclusive

6.12 NAK or Reply to Invalid Property Exchange inquiry

To NAK an Inquiry or if an invalid Inquiry is received, respond by sending the associated Reply message with values of the message fields as follows:

Set the Number of Chunks, Number of This Chunk, and Length of Following Property Data to zero. Do not include any other payload before the F7.

Value	Parameter
F0	System Exclusive Start
7E	Universal System Exclusive
1 byte	Source 7F = to/from whole MIDI Port 00-0F = to/from MIDI Channels 1-16
0D	Universal System Exclusive Sub ID #1: MIDI-CI
1Byte	Universal System Exclusive Sub ID #2: 0x31 Reply to Property Exchange Capabilities 0x33 Reply to Has Property 0x35 Reply to Get Property 0x37 Reply to Set Property
00	MIDI-CI Message Version/Format
4 bytes	MNID (MIDI Negotiation Identifier)
00 00	Number of Chunks in Data Set
00 00	Number of This Chunk
00 00 00 00	Length of Following Property Data
F7	End Universal System Exclusive

6.13 General Mechanisms of Property Exchange

Property Exchange messages may be used independently. There is no defined message flow (other than inquiry generally prompts an associated reply). However, a common exchange might typically begin as in the following example.

Example Sequence to Discover a Property

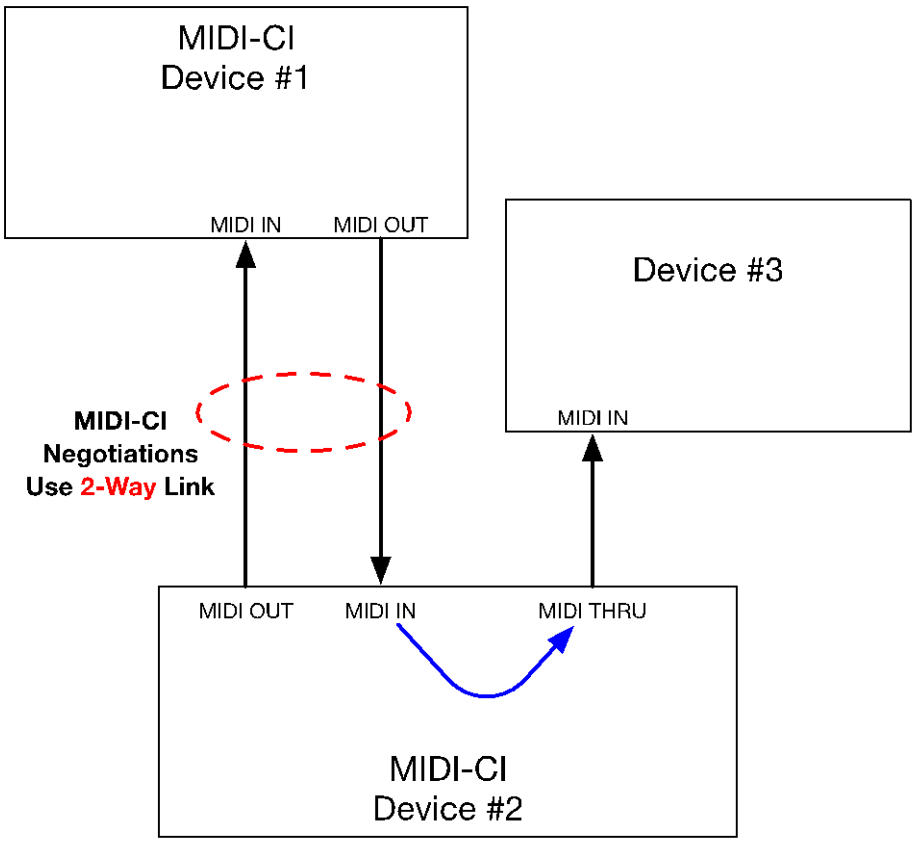
Step	Device	Request Type	Action
1	Initiator	0x30 Inquiry Property Exchange Capabilities	Query basic capabilities of Property Exchange. (This inquiry does not need to be performed for every session.)
2	Responder	0x31 Reply Property Exchange Capabilities	Reply with basic capabilities for Property Exchange.
3	Initiator	0x32 Inquiry Has Property	Query whether a receiver has the specified Property and whether it is possible to set the value.
4	Responder	0x33 Reply to Has Property	Reply whether device has the specified Property (true / false) and whether it is possible to set the value (true / false).
5	Initiator	0x34 Inquiry Get Property	Query the specified Property value
6	Responder	0x35 Reply to Get Property	Returns the value of the specified Property*

*If invalid Property is specified, NAK is returned.

Appendix A: MIDI Thru Limitation

It is strongly recommended not to use MIDI Thru ports when using MIDI-CI.
Behavior is undefined.

MIDI-CI negotiations will work reliably on a bidirectional link between 2 devices that both support MIDI-CI. MIDI-CI does not support all topologies that MIDI 1.0 allows. In particular, MIDI Thru ports add a complication that might cause **significant errors**.



Device #3 does not have the 2-way connection necessary for MIDI-CI. It may not be able to support changes that are negotiated between Device #1 and Device #2. Behavior is undefined.

Future MIDI-CI devices might provide an intelligent MIDI Thru function that performs any conversions necessary to support a MIDI device connected by the MIDI Thru port to help mitigate potential incompatibility issues for the user. The details of such a design are not defined in this version of MIDI-CI.

Appendix B: List of All MIDI-CI Messages

List of all MIDI-CI Messages sorted by Universal System Exclusive Sub ID #2

Universal System Exclusive Sub ID #2: Type of MIDI-CI Message
0x00-0F Reserved
0x10-1F Protocol Negotiation Messages
0x20-2F Profile Configuration Messages
0x30-3F Property Exchange Messages
0x40-7E Reserved
0x7F NAK

Sub ID #2	Message Type
Reserved Space	
0x00-0x0F	Reserved
Protocol Negotiation Messages	
0x10	Initiator: Protocol Inquiry & Report Protocol Capabilities
0x11	Responder: Report Protocol Capabilities
0x12	Initiator: Set New Selected Protocol
0x13	Initiator: Test New Protocol Initiator to Responder
0x14	Responder: Test New Protocol Responder to Initiator
0x15	Initiator: Confirmation Protocol Established
0x16	Reserved
0x17	Reserved
0x18	Reserved
0x19	Reserved
0x1A	Reserved
0x1B	Reserved
0x1C	Reserved
0x1D	Reserved
0x1E	Reserved
0x1F	Reserved
Profile Configuration Messages	
0x20	Profile Inquiry
0x21	Responder Reply Profile Capabilities
0x22	Set Profile On
0x23	Set Profile Off
0x24	Profile Enabled Report
0x25	Profile Disabled Report
0x26	Reserved
0x27	Reserved
0x28	Reserved
0x29	Reserved

0x2A	Reserved
0x2B	Reserved
0x2C	Reserved
0x2D	Reserved
0x2E	Reserved
0x2F	Reserved
Property Exchange Messages	
0x30	Inquiry Property Exchange Capabilities
0x31	Reply to Property Exchange Capabilities
0x32	Inquiry Has Property
0x33	Reply to Has Property
0x34	Inquiry Get Property
0x35	Reply to Get Property
0x36	Inquiry Set Property
0x37	Reply to Set Property
0x38	Reserved
0x39	Reserved
0x3A	Reserved
0x3B	Reserved
0x3C	Reserved
0x3D	Reserved
0x3E	Reserved
0x3F	Notify Message
Reserved Space	
0x40-7E	Reserved
Management Messages	
0x7F	NAK

